RCTG

RECURSIVE COGNITIVE TASK GRAPH

Visual Architecture Reference

Every component. Every connection. One IP address. The terrifying output.

1. Master System Architecture

This is the full picture. Every service, every database, every agent — all wired together. The Opportunity Tree is the center of gravity. Everything else either feeds it, reads from it, or executes on behalf of it.

Three things to read in this diagram before anything else:

- Everything flows back to the Opportunity Tree — agents write to nodes, nodes spawn agents, the rescore agent updates nodes. The tree never stops growing.
- Storage is split by purpose — PostgreSQL for structure, Neo4j for relationships, Redis for speed. No single database is asked to do everything.
- Human-in-the-Loop sits above the scheduler as a preemptive interrupt — not a step in a workflow.

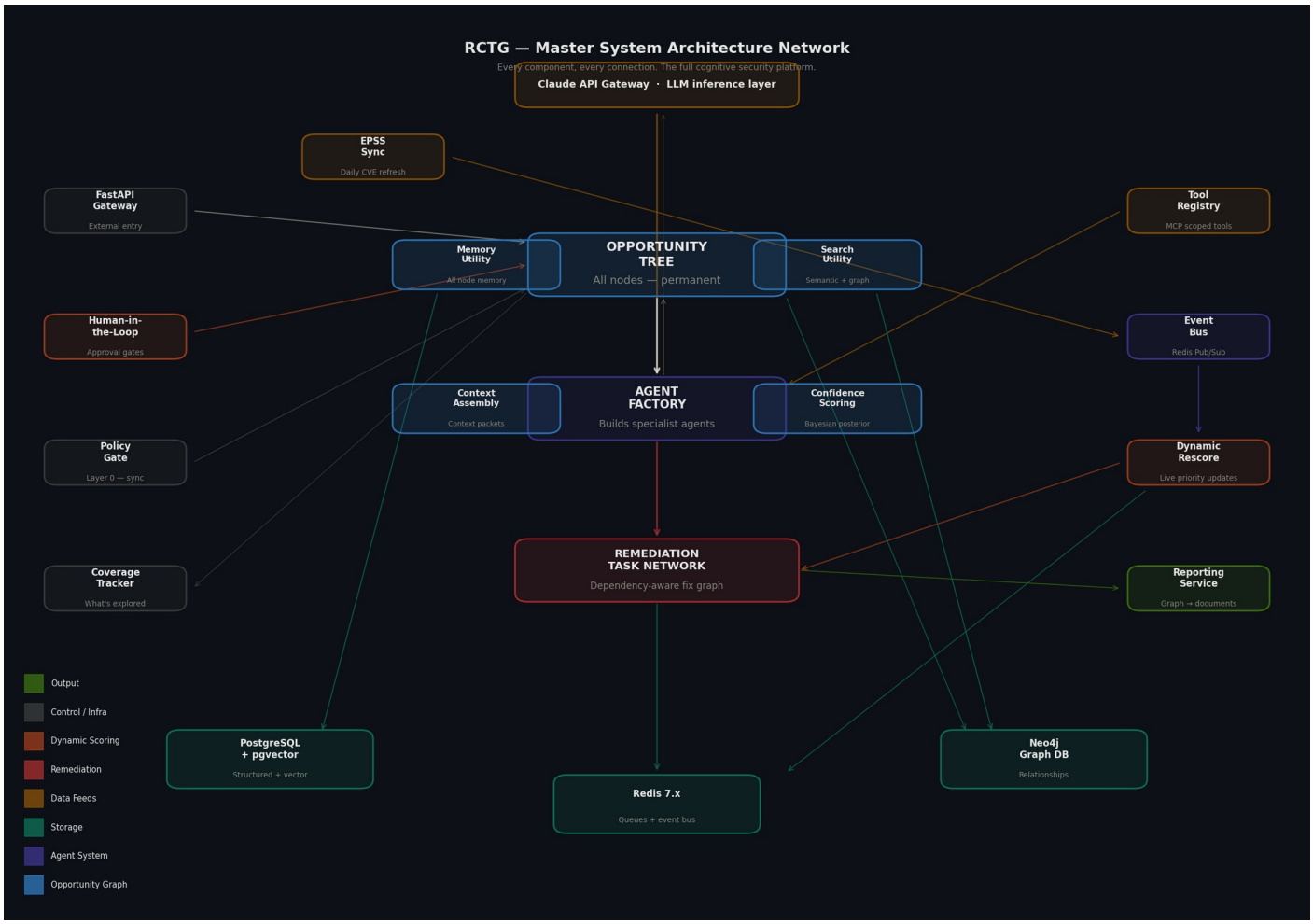

Figure 1 — Master system network: all components and their connections

The graph is the brain. Not the agent. The agent is disposable. The graph accumulates everything permanently.

2. The Opportunity Tree

The opportunity tree is a graph that grows and never shrinks. Every node has a type, a stage, a status, a memory, a priority score, and a set of children. When an agent completes, it writes findings to the node and creates child nodes. Those children get scored, queued, and have their own agents spawned. The cycle runs indefinitely.

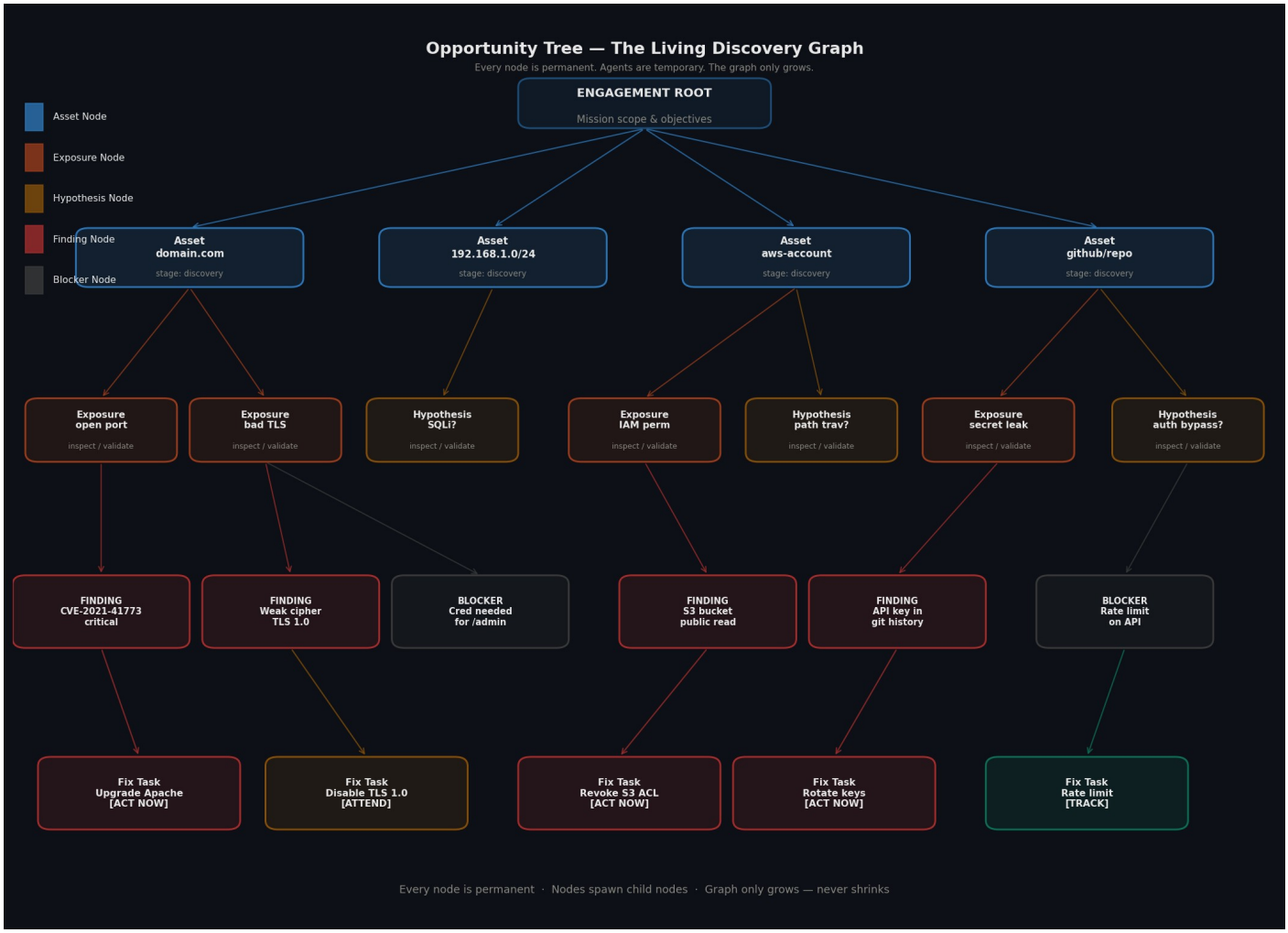

Figure 2 — Opportunity tree: all five node types, investigation stages, and how the graph expands root to fix tasks

The five node types: Asset (the thing being investigated), Exposure (something observed that could be a problem), Hypothesis (a testable idea with a safe validation method), Finding (a confirmed, evidenced issue), and Blocker (a dependency preventing investigation — resolved as a side quest).

Nodes are permanent. Never deleted. Only archived. Historical state is always recoverable. An agent that produced a false positive yesterday is an audit record, not a missing data point.

3. Agent Factory

The agent factory is a 10-step sequential build process. Every step is logged as an immutable audit record before a single token goes to the LLM. Steps 1 through 5 run on the left column: Policy Gate → Capability Analyzer → Tool Selector → Budget Planner → Context Assembly. Steps 6 through 10 run in parallel on the right: Prompt Builder → Validator Builder → Metrics Builder → AgentRun Record (created BEFORE spawn) → Agent Spawned.

Figure 3 — Agent factory: 10-step sequential build process from node selection to agent spawn

The AgentRun database record is created at Step 9 — BEFORE the agent spawns. The audit trail begins before execution, not after. This is non-negotiable.

4. Scheduler — 6-Layer Interrupt Model

The scheduler runs every 5 seconds as a dedicated async service. Six layers run in strict order every cycle. Higher layers preempt lower layers completely. The queues are Redis sorted sets — ZPOPMAX pulls the highest-score node first. The scheduler outputs action verbs (ACT NOW, ATTEND, TRACK, DEFER), not raw scores.

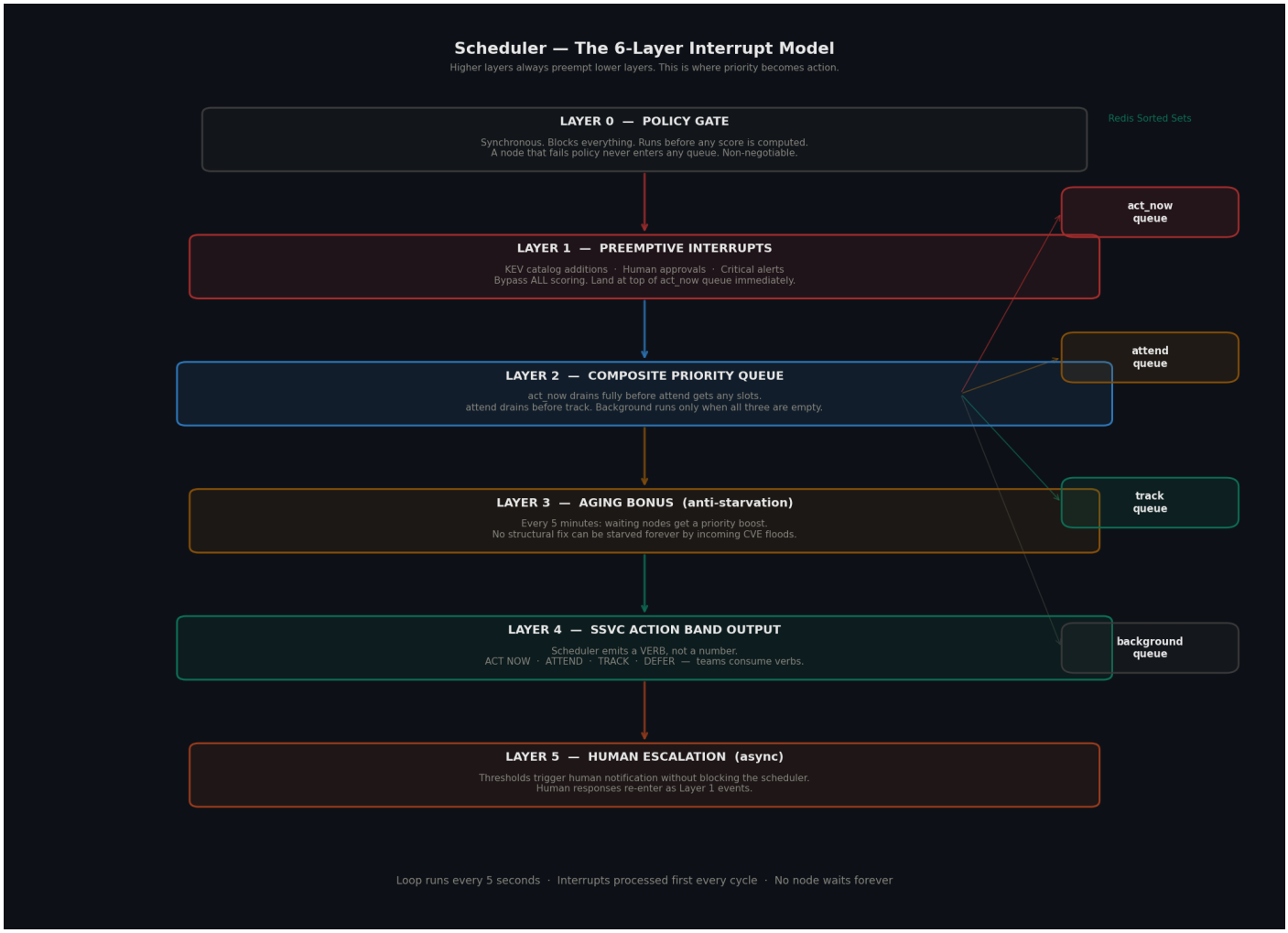

Figure 4 — Scheduler: 6 interrupt layers with Redis sorted set queues on the right

The aging bonus (Layer 3) prevents starvation — the same problem solved in OS scheduling theory in the 1970s. Without aging, a flood of new critical CVEs will perpetually displace structural fix tasks. With aging, every waiting node eventually bubbles up.

5. Priority Engine — Composite Scoring

EPSS gets the highest weight at 40% because exploit probability is the strongest real-world predictor of what attackers actually target. CVSS measures severity — how bad if exploited. EPSS measures likelihood — will it actually be exploited. These are different questions requiring different inputs. CVSS alone at 30% weight is not enough to drive prioritization.

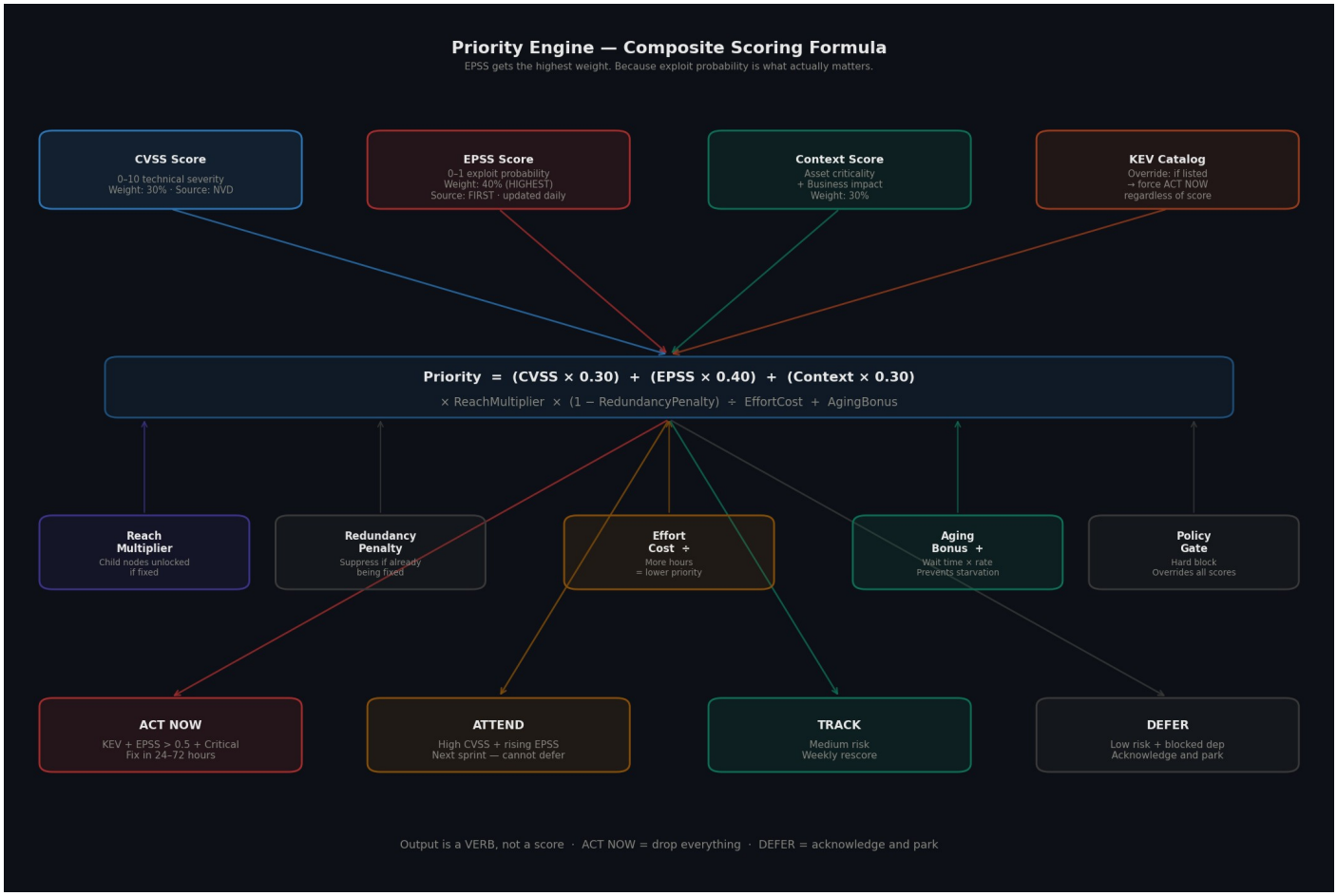

Figure 5 — Priority engine: composite formula, input nodes, modifiers, and SSVC output bands

Priority = (CVSS × 0.30) + (EPSS × 0.40) + (Context × 0.30) × ReachMultiplier × (1 − RedundancyPenalty) ÷ EffortCost + AgingBonus

The output is a verb, not a number. ACT NOW means drop everything. ATTEND means schedule this sprint. TRACK means watch weekly. DEFER means acknowledge and park. Teams consume the verb. The score is internal.

6. Universal Utility Layer — 20 Shared Services

The utility layer is the nervous system. Every agent, every scheduler cycle, every factory build draws from these 20 services. Without the utility layer, every agent would need to implement its own memory lookup, its own confidence scoring, its own relationship tracking — and they would all disagree. The utility layer ensures every part of the system operates on one consistent view of reality.

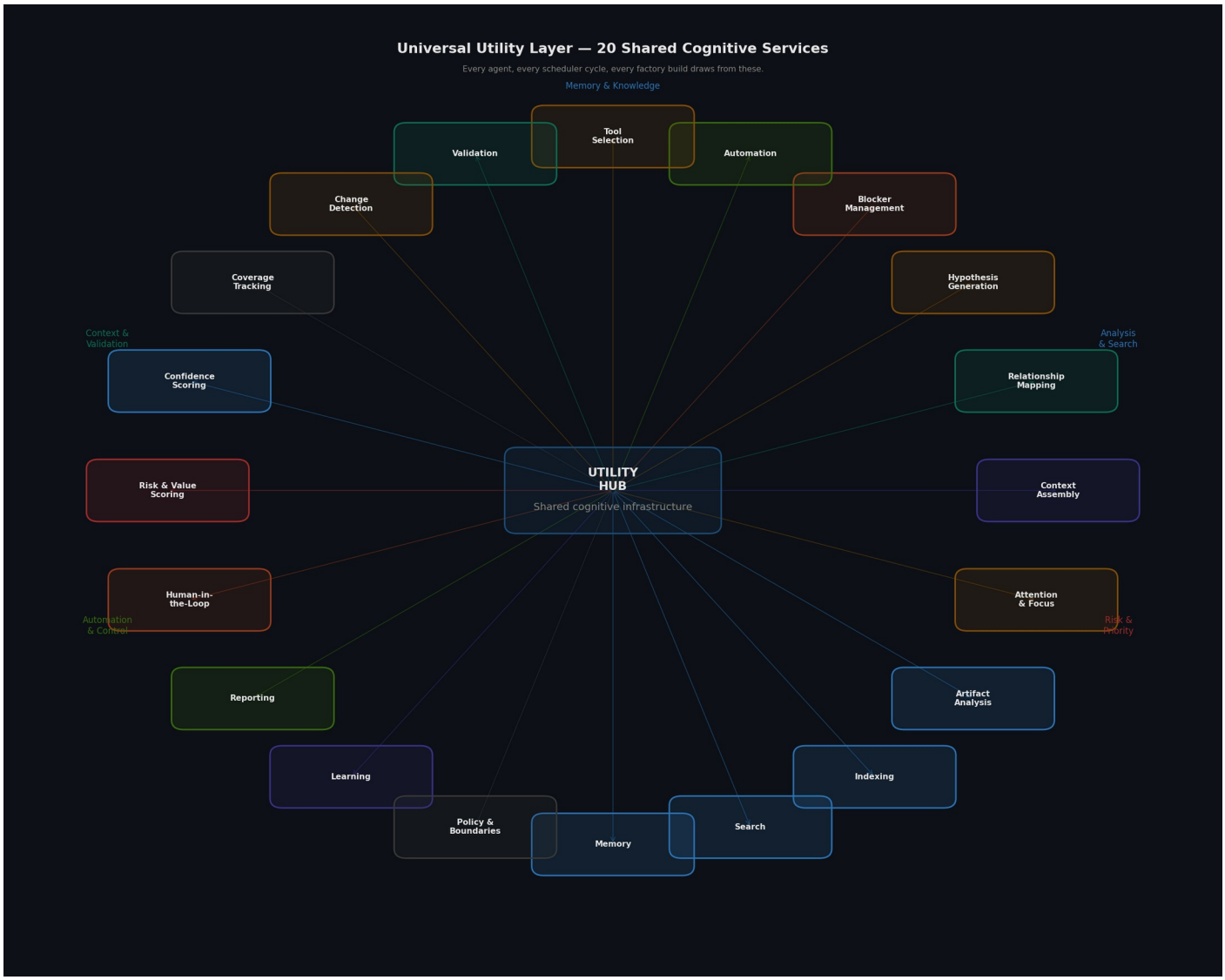

Figure 6 — Utility layer: 20 shared services in a radial network around the central hub

7. Individual Utility Deep Dives

7.1 Memory Utility

Five memory layers with different storage backends, different access patterns, and different confidence decay rates. Cloud/IAM facts decay 10× faster than DNS records because they change constantly. Any fact below 0.65 confidence triggers a revalidation — a new agent is spawned to re-verify before the fact can be cited as evidence.

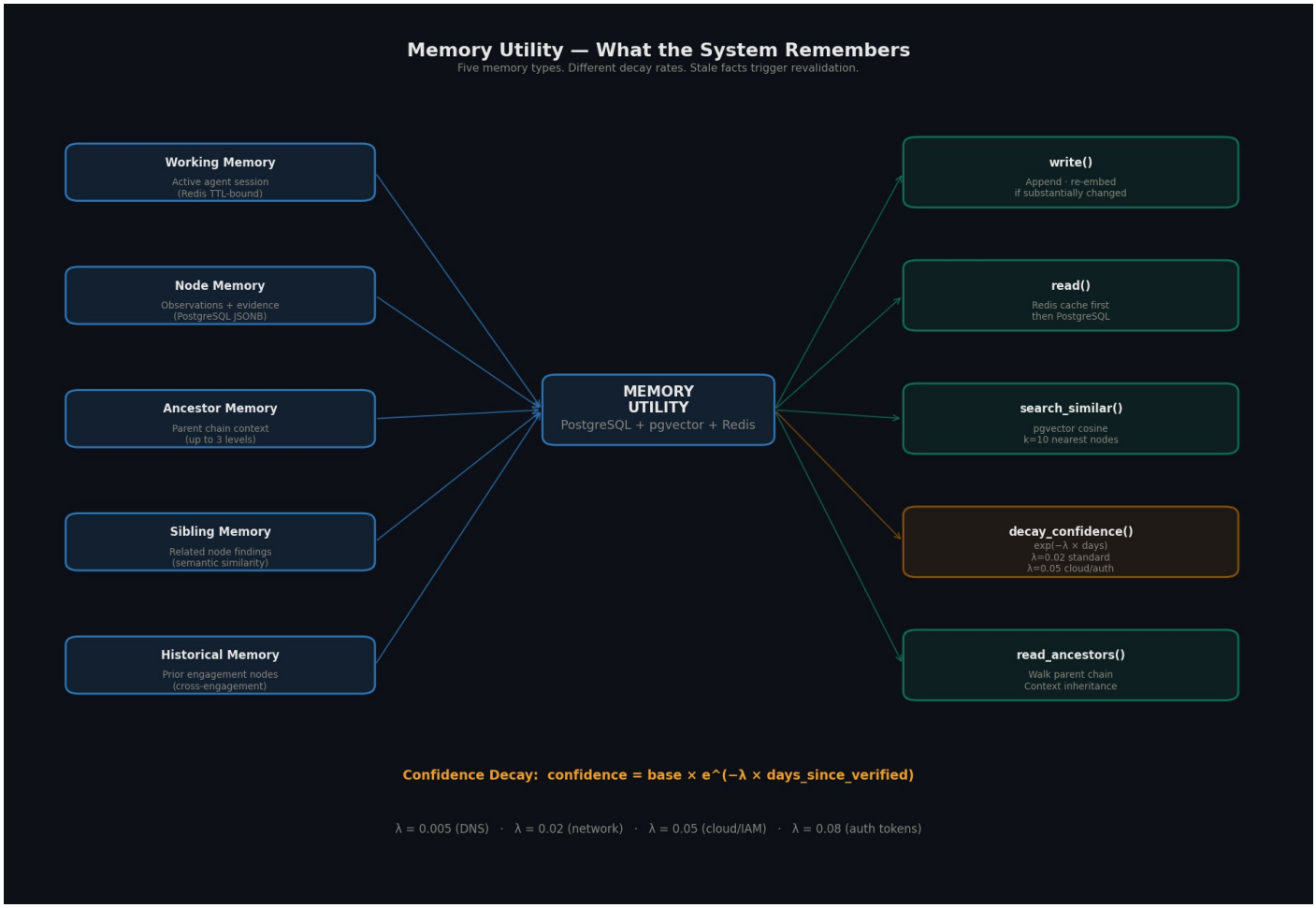

Figure 7 — Memory utility: five memory types, operations, and confidence decay formula

7.2 Confidence Utility

Every fact has a Bayesian posterior confidence score. Not high/medium/low — an actual number between 0 and 1. Three aggregation methods handle different evidence scenarios: noisy-or when any source being right is sufficient, minimum for chains where all steps must be reliable, and mean for corroborating evidence.

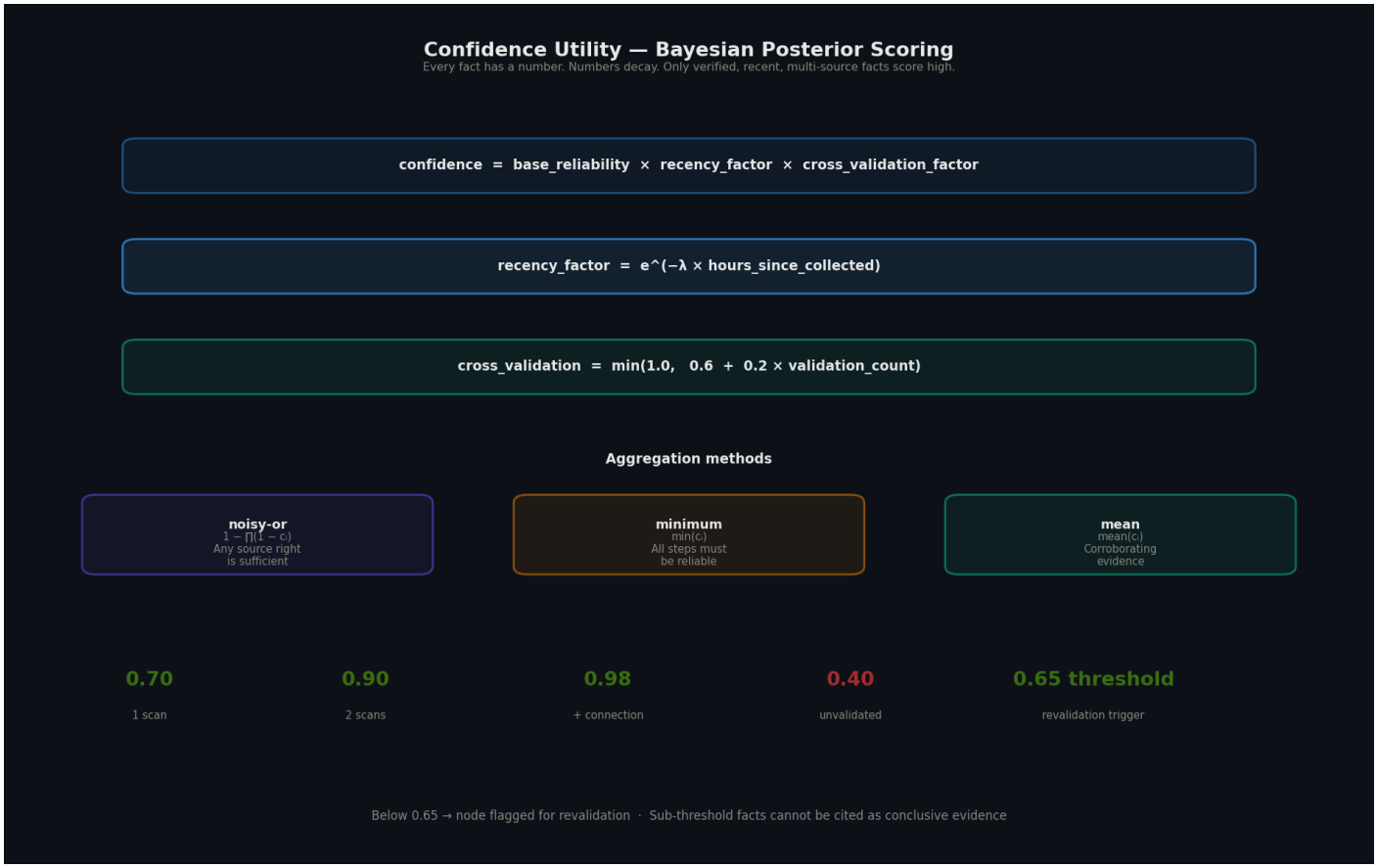

Figure 8 — Confidence utility: Bayesian scoring formula, aggregation methods, example scores

7.3 Context Assembly Utility

Every agent receives a carefully constructed context packet assembled from six sources. When the packet exceeds the 32,000 token budget, it trims in strict priority order — current node memory and constraints are NEVER trimmed. Historical context and distant siblings go first. The agent always knows who it is and what it cannot do.

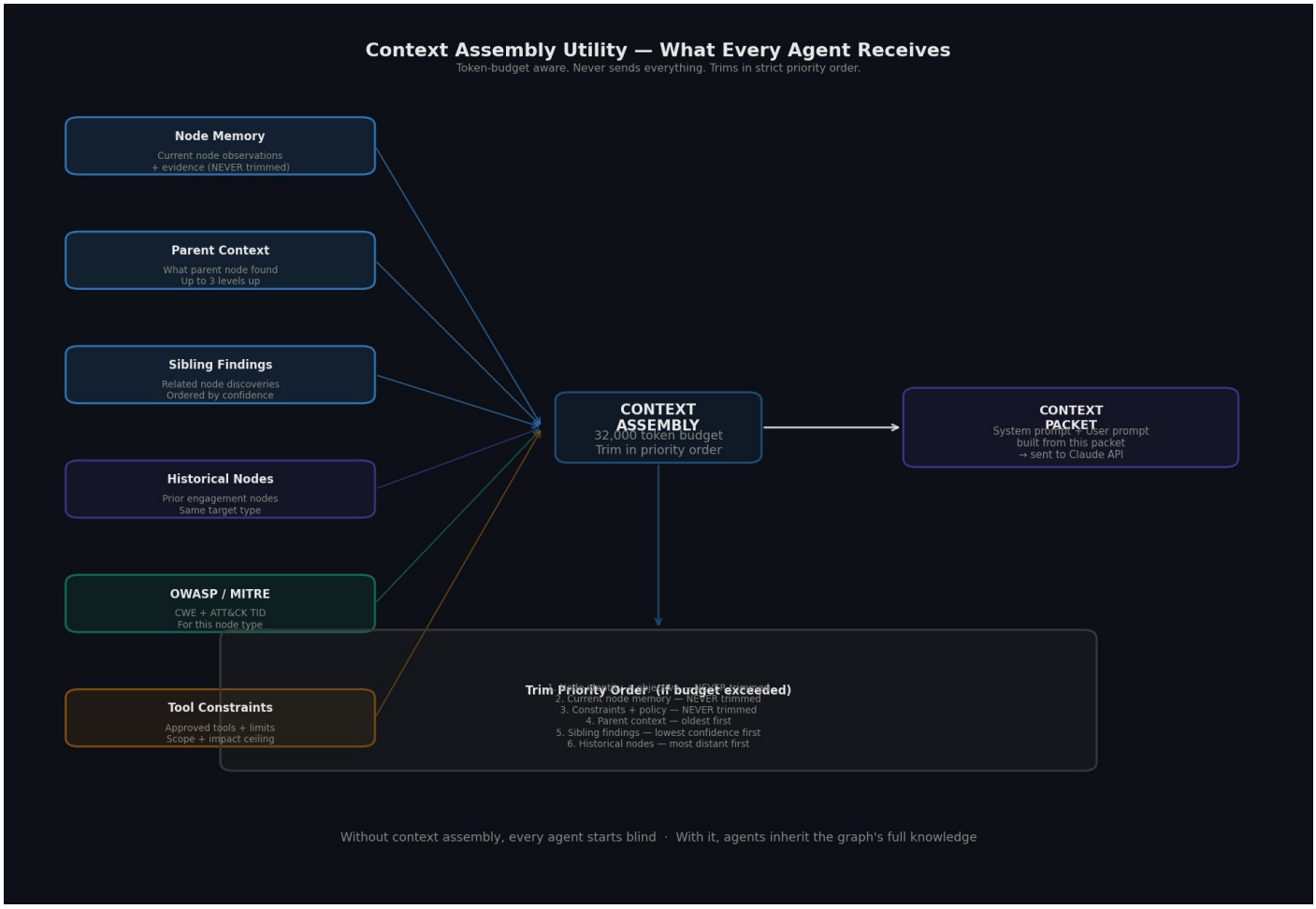

Figure 9 — Context assembly: six input sources, trim priority order, output packet to LLM

7.4 Relationship Mapping Utility

Every discovered relationship becomes a typed, directed edge in Neo4j. The system doesn't just store that Apache 2.4.51 is vulnerable to CVE-2021-41773 — it stores the entire chain from host to service to CVE to finding to fix task to retest node. Graph traversal makes that chain queryable in milliseconds.

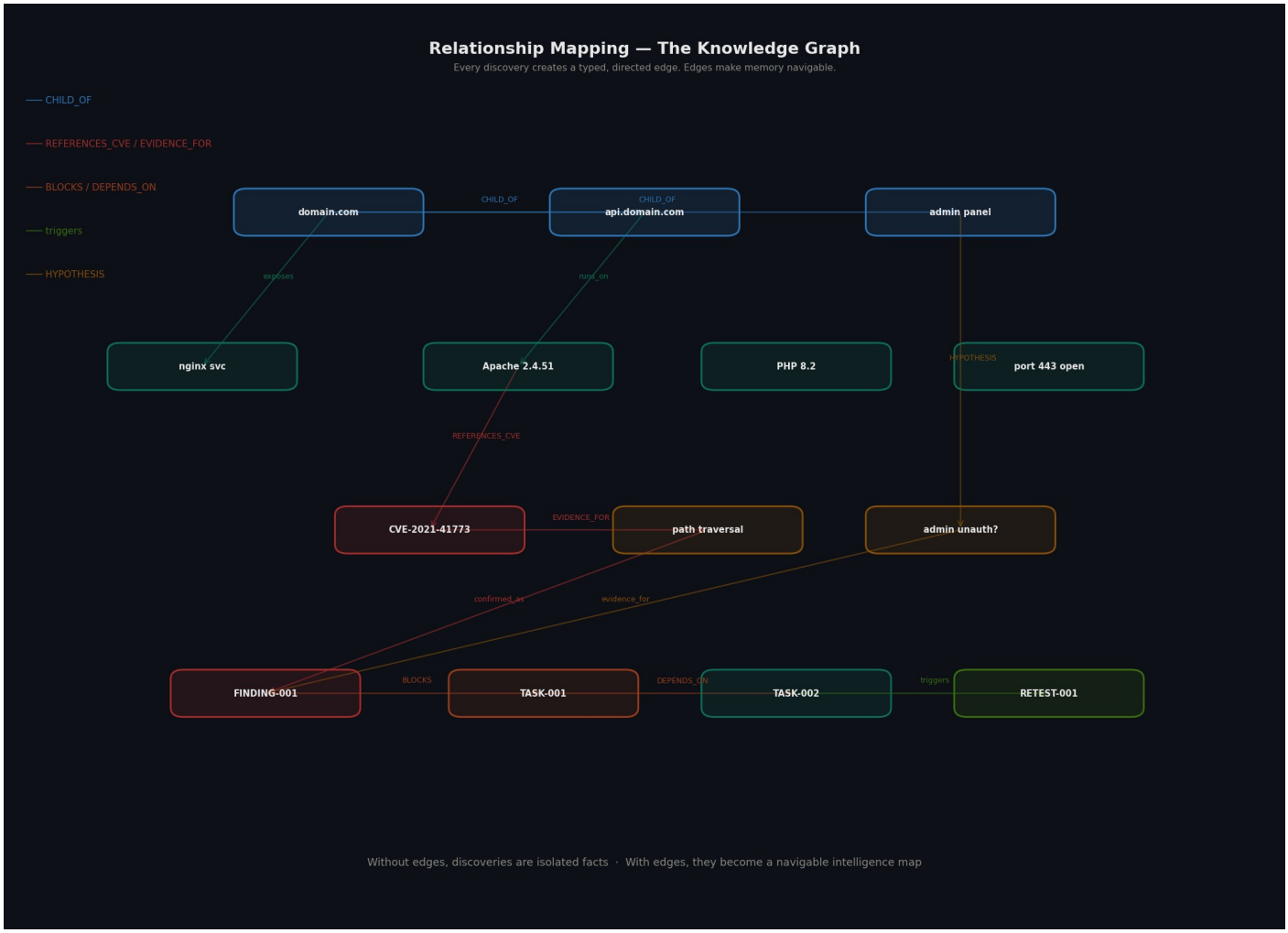

Figure 10 — Relationship mapping: knowledge graph with typed edges and provenance

7.5 EPSS Sync + KEV Monitor

Runs at 02:00 UTC daily. Pulls every EPSS score for every published CVE. Diffs against yesterday. If a score changed by more than 5%, publishes a rescore event. If a CVE is newly added to the CISA KEV catalog, it goes directly to the interrupt queue — forcing ACT NOW on every affected node regardless of composite score.

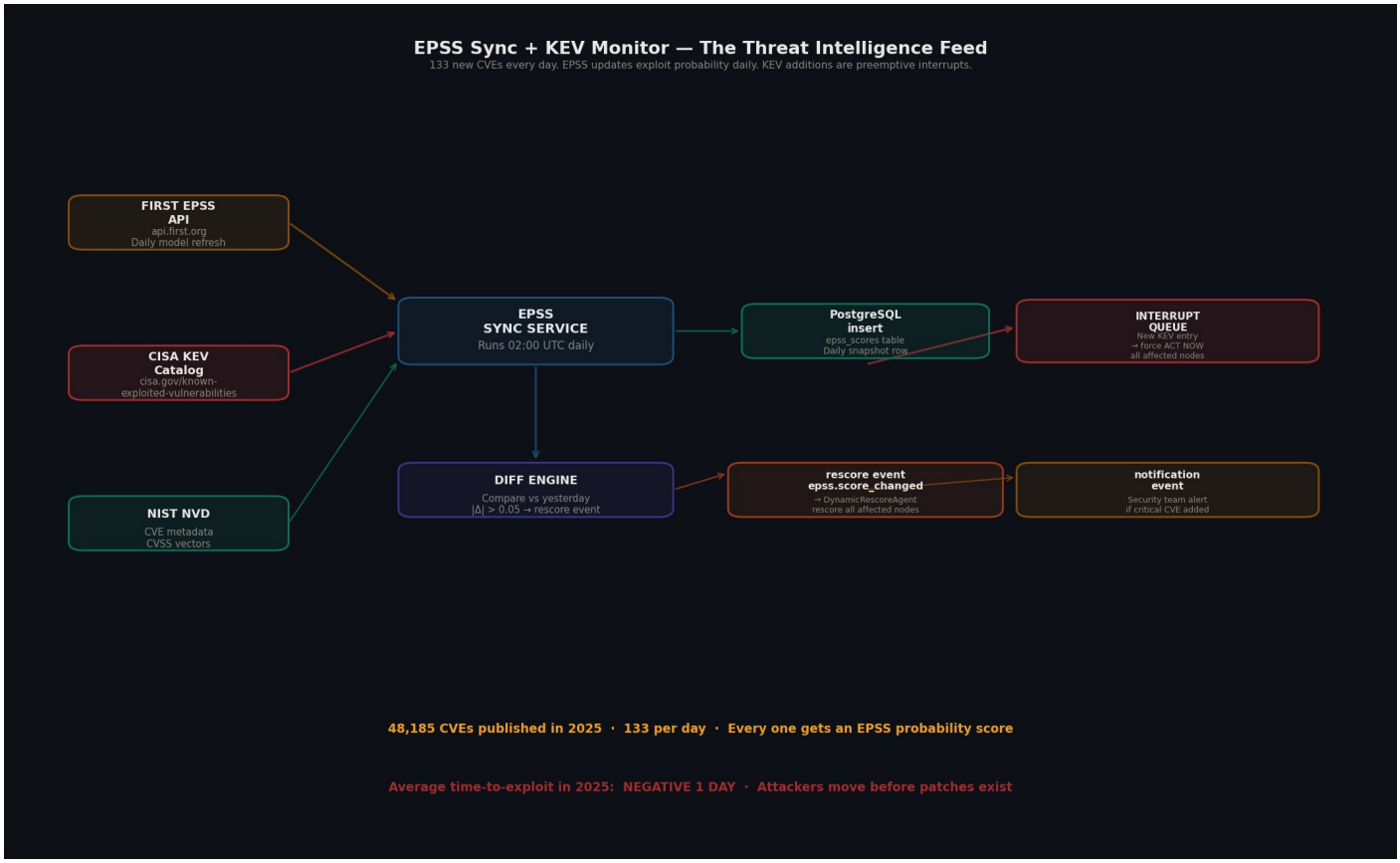

Figure 11 — EPSS sync network: daily refresh, diff engine, rescore events, KEV interrupts

7.6 Human-in-the-Loop Gates

HIL is a preemptive interrupt, not a workflow step. Five conditions fire it: high blast radius (CAB required), owner dispute, regression risk, SLA breach imminent, or low agent confidence on a critical node. The blocked node sits in holding state until a human responds. The response re-enters as a Layer 1 interrupt event.

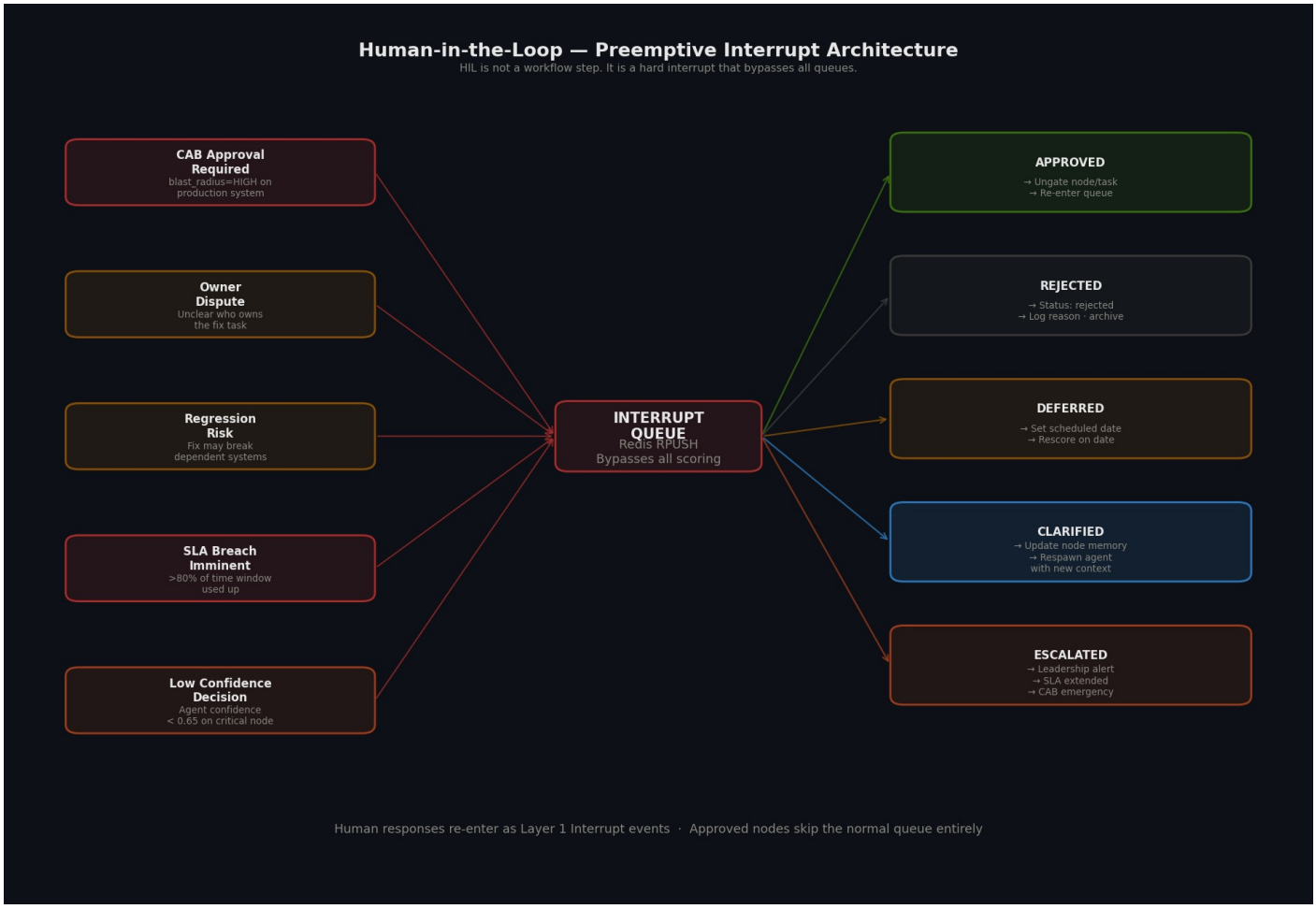

Figure 12 — Human-in-the-loop: trigger conditions, interrupt architecture, five response types

8. Storage Architecture

Three databases, each chosen for what it actually does best. Neo4j handles graph traversal — finding all nodes transitively blocked by a blocker in a 1000-node graph — in milliseconds. The equivalent SQL JOIN would be unacceptably slow. PostgreSQL handles ACID writes, time-series EPSS data, and vector similarity via pgvector. Redis handles speed and events.

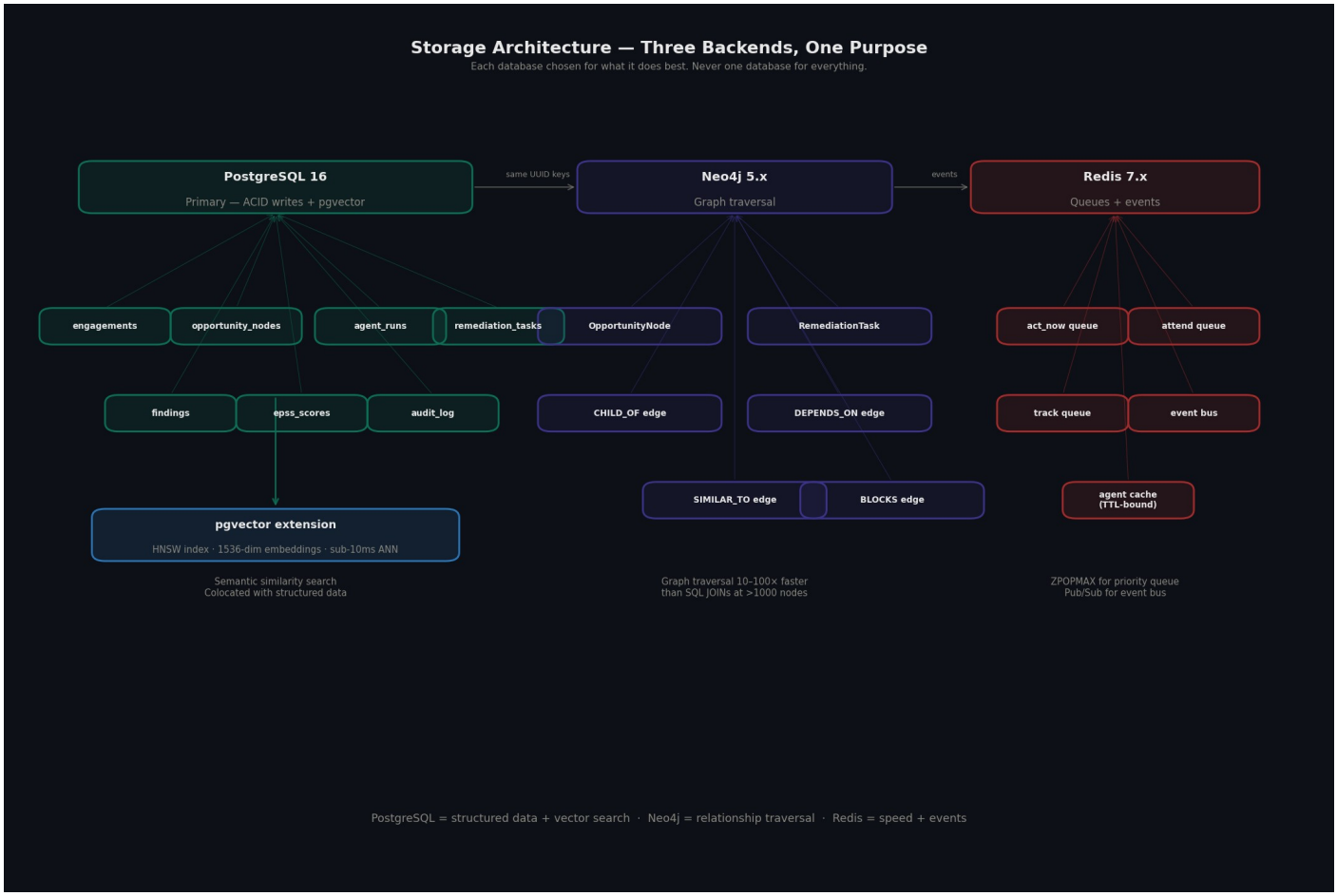

Figure 13 — Storage: PostgreSQL + pgvector, Neo4j, Redis — data domains and cross-connections

9. Remediation Task Network

Every fix task carries full metadata: owner, dependencies, blast radius, approval tier, SLA date. The dependency graph between tasks has real-world consequences — patching the app before patching the OS it runs on may invalidate the patch. Kahn's topological sort runs over the DEPENDS_ON edges to produce the correct execution order.

The Dynamic Rescore Agent runs continuously. Four events trigger immediate rescoring: EPSS shift greater than 5%, new KEV entry (preemptive interrupt), new finding injected, or fix task deployed (unblocks its dependents). The network is alive — it updates without anyone touching it.

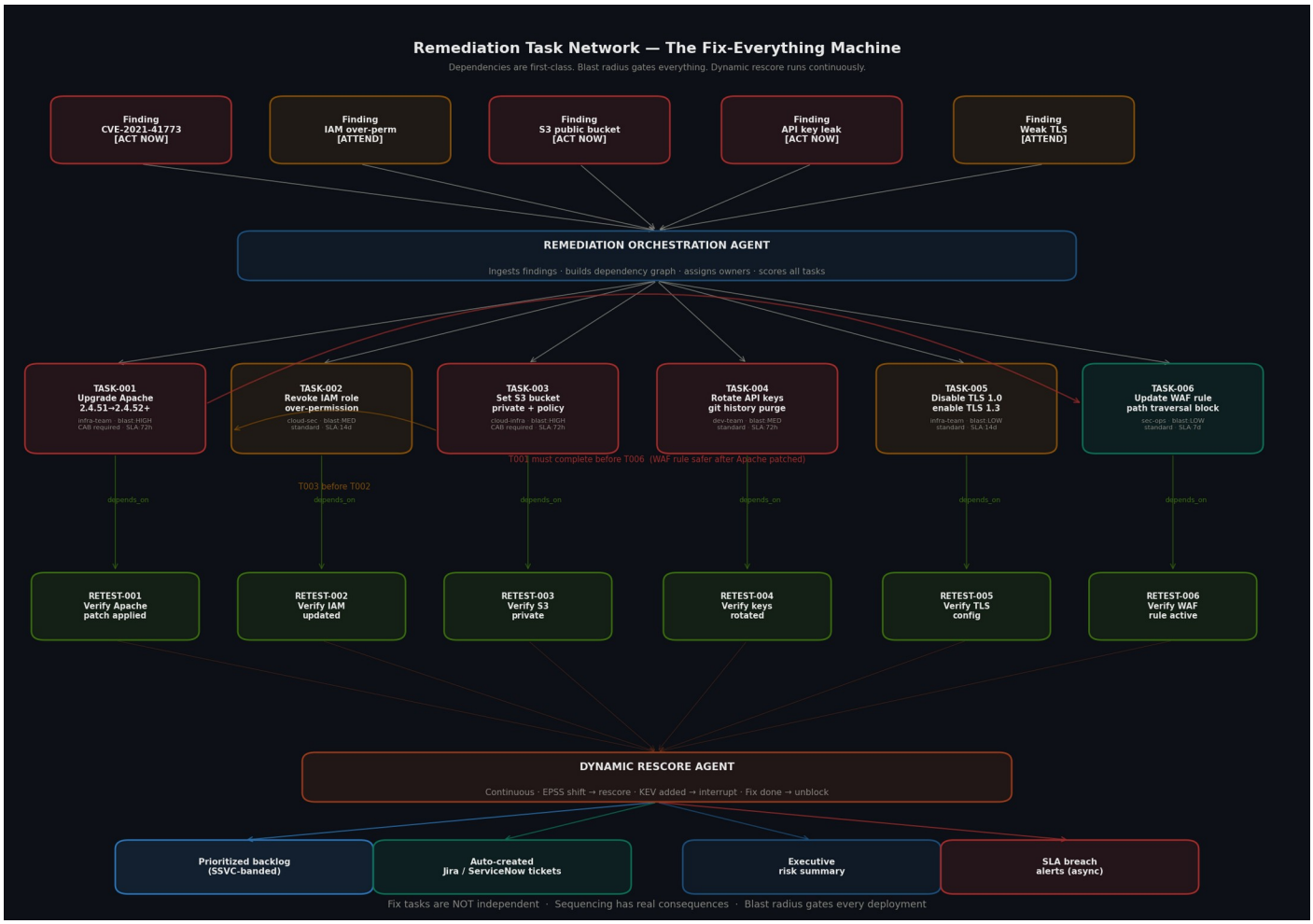

Figure 14 — Remediation task network: findings to tasks to retests with dependency arrows and dynamic rescore

10. The Walkthrough — 192.168.1.105

A single IP address. That is all we give the system. What follows is step-by-step exactly what happens — with theoretical tool outputs, actual CVE data, and real EPSS scores.

Every CVE used in this walkthrough is real. CVE-2021-41773 (Apache path traversal) has an EPSS score of 0.974 — the FIRST model estimates a 97.4% probability of exploitation in the next 30 days. CVE-2020-1938 (Tomcat Ghostcat) has a CVSS of 9.8. MySQL 5.7 reached end of life in October 2023 — zero patches available. PHP 7.2 reached end of life in December 2019. These are not edge cases. This is typical enterprise infrastructure.

Stage 0 — Input and Node Creation

The IP is entered via API. Scope check passes. A root asset node is created with a default score of 0.50 and placed in the ATTEND queue. The scheduler picks it up within 5 seconds of the next cycle.

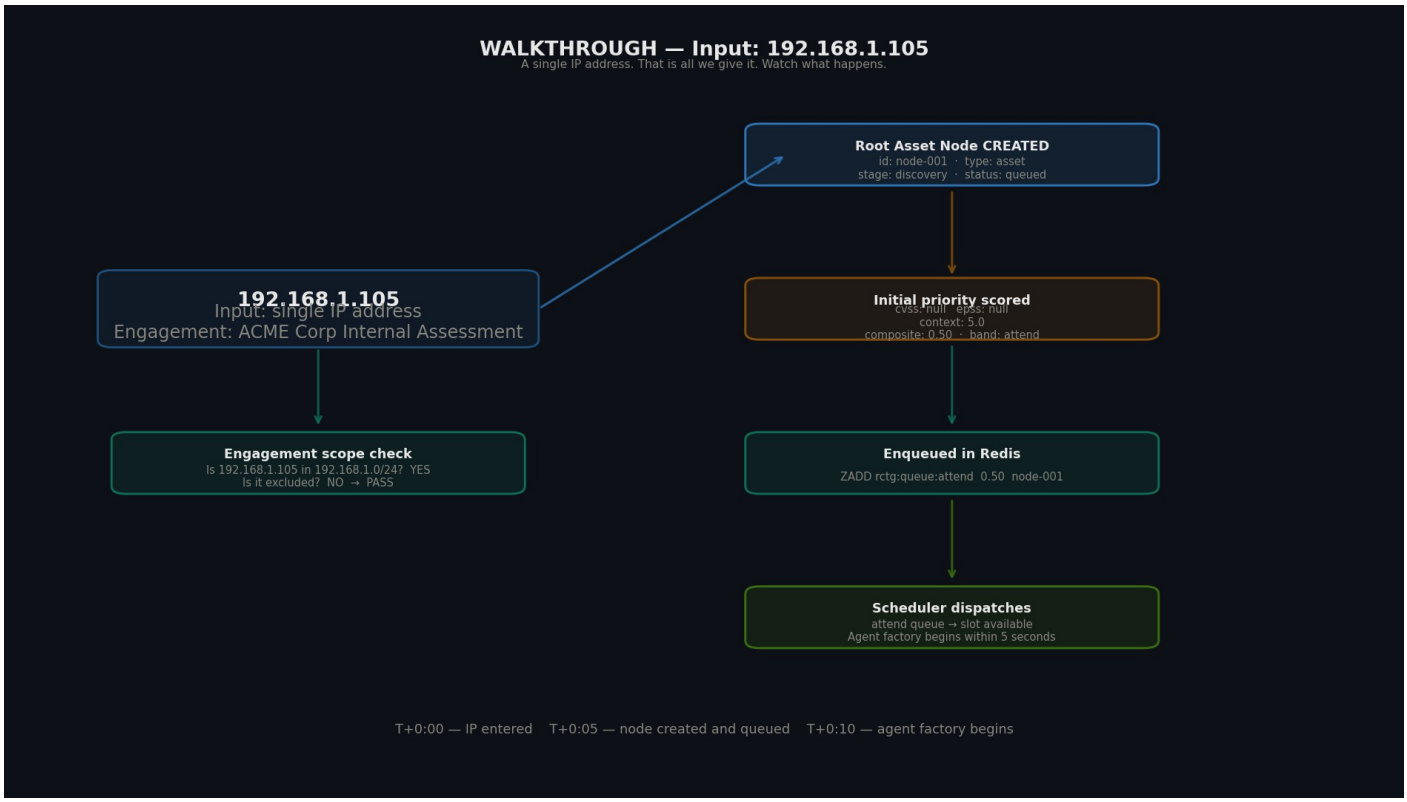

Figure 15 — Stage 0: IP input → scope check → root node created → queued → dispatched

Stage 1 — Discovery Agent

The factory builds an Asset Discovery Agent. Nine tool calls execute: port scan reveals 22, 80, 443, 3306, 8080, 8443, 5432. Banner grabs return Apache 2.4.51 with PHP/7.2 header, MySQL

5.7.38, OpenSSH 7.4. HTTP probes find a self-signed SSL cert expired in January 2022 (three years overdue), Tomcat 8.5.73 with /manager accessible, and an admin interface on port 8080. DNS lookup resolves to db-internal-01.acme-corp.local — a database server. Confidence: 0.91.

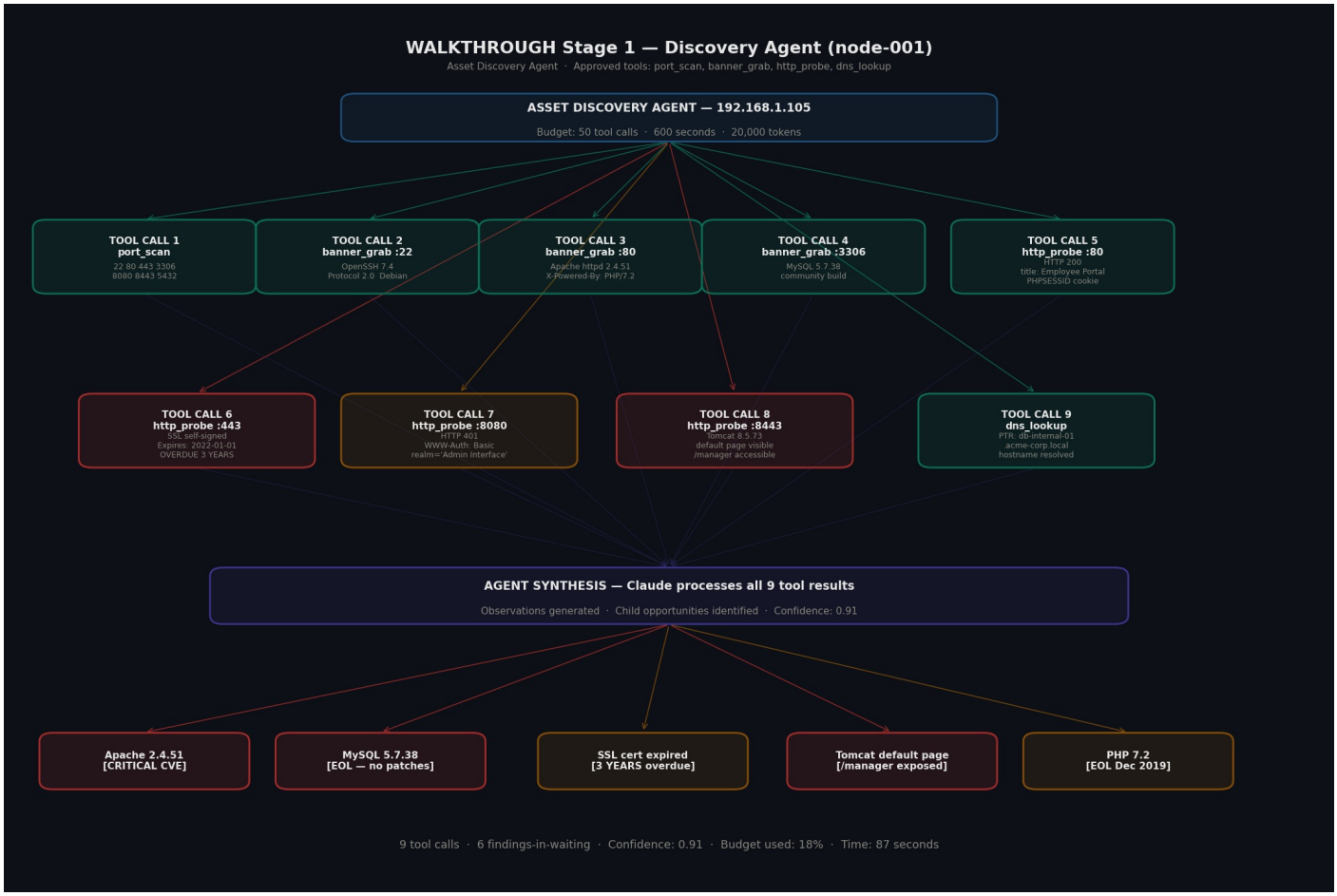

Figure 16 — Stage 1: discovery agent executes 9 tool calls, identifies 5 critical findings-in-waiting

This is one server. Running 5 end-of-life or critically vulnerable software components simultaneously. This is not unusual. This is typical of enterprise infrastructure without continuous security assessment.

Stage 2 — Child Node Explosion

Eight child nodes are created simultaneously and scored by the priority engine. Three land in ACT NOW immediately: node-002 (Apache CVE-2021-41773, EPSS 0.974, score 2.68), node-006 (Tomcat Ghostcat CVE-2020-1938, score 2.45), node-008 (Tomcat /manager default creds hypothesis, score 2.31). The scheduler drains ACT NOW completely before attend or track get any worker slots.

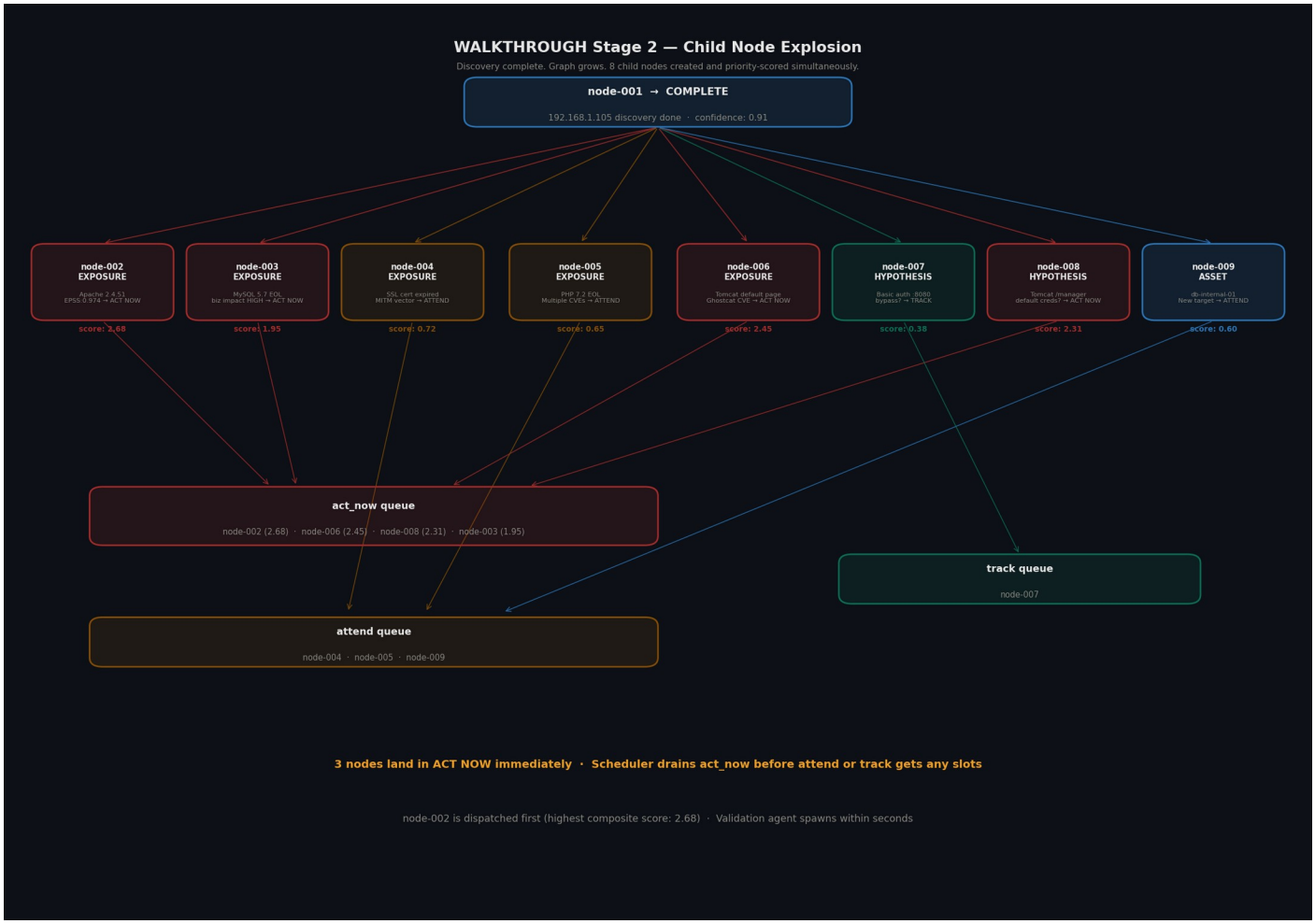

Figure 17 — Stage 2: 8 child nodes created, scored, and routed to their priority queues

Stage 3 — Validation Confirms the Critical

A Hypothesis Validation Agent spawns for node-002. Its only approved tool is http_request with read-only constraint. Step 1 confirms the Apache version. Step 2 runs a single safe PoC probe: HTTP GET /cgi-bin/.%2e/.%2e/etc/passwd. The server returns HTTP 200 with the contents of /etc/passwd. The vulnerability is confirmed in under 2 minutes. Confidence: 0.99.

The validator confirms: evidence quality HIGH, scope CLEAR, false positive near-zero, blast radius HIGH. A Finding node is created. The KEV check fires — CVE-2021-41773 is on the catalog. A preemptive interrupt goes to the queue. CAB gate opens. Human escalation fires.

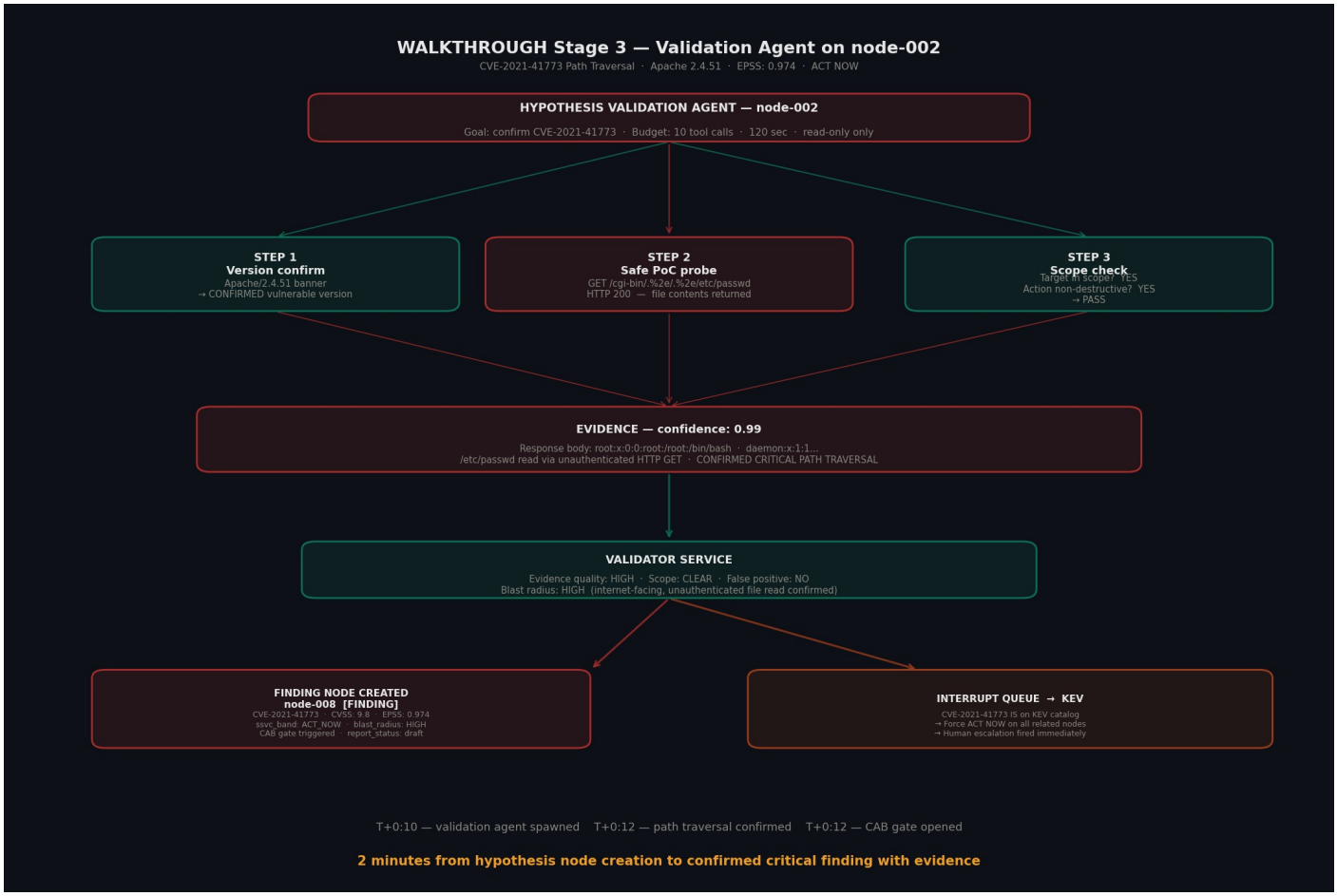

Figure 18 — Stage 3: validation agent confirms path traversal, KEV interrupt fires, CAB gate triggered

T+0:12. 12 minutes from IP entry to confirmed critical RCE vulnerability with evidence. The equivalent manual process takes days to weeks.

Complete Timeline

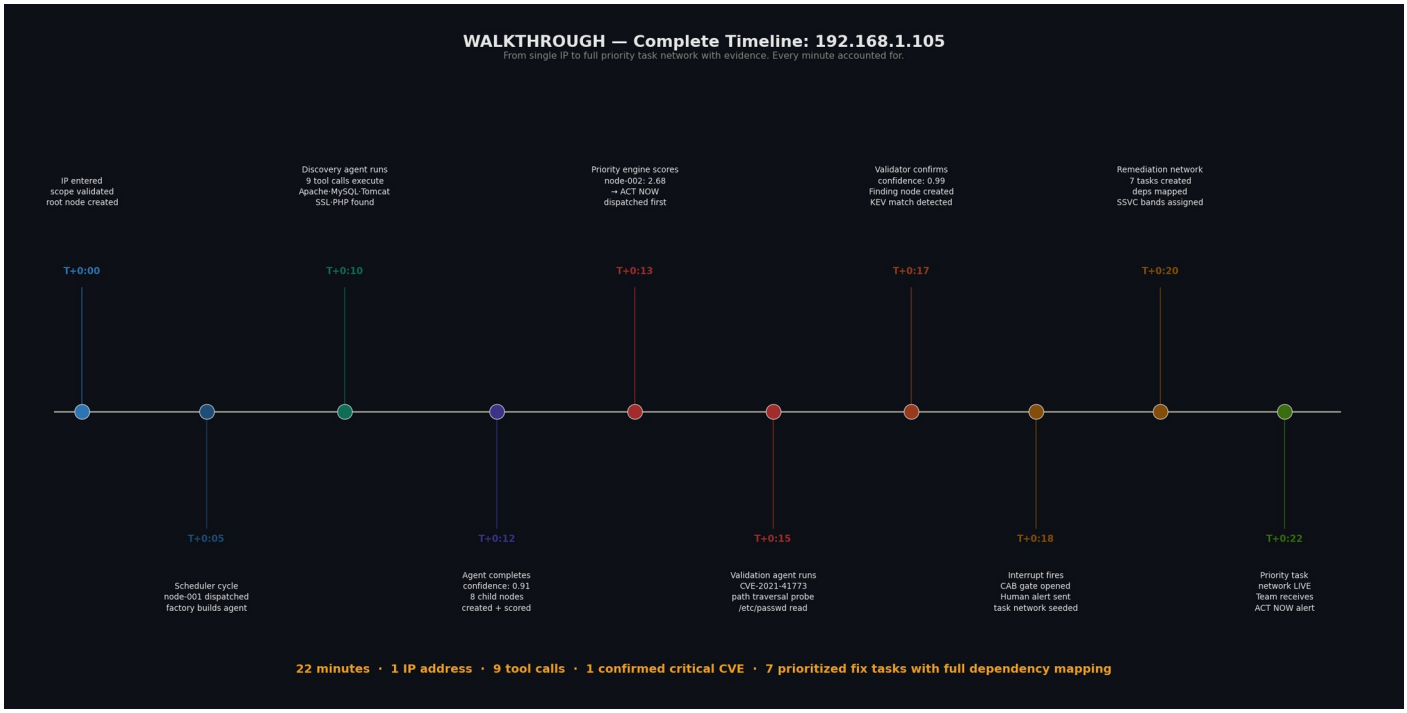

Figure 19 — Complete timeline: T+0:00 to T+0:22, all 10 events

11. The Output — Priority Task Network

This is what the security team receives. Not a PDF. A live, prioritized, dependency-aware action network. Three tasks in ACT NOW. Four in ATTEND. One in TRACK. Five retests pre-created and blocked until fixes deploy. A new asset discovered (db-internal-01) that now needs its own discovery run.

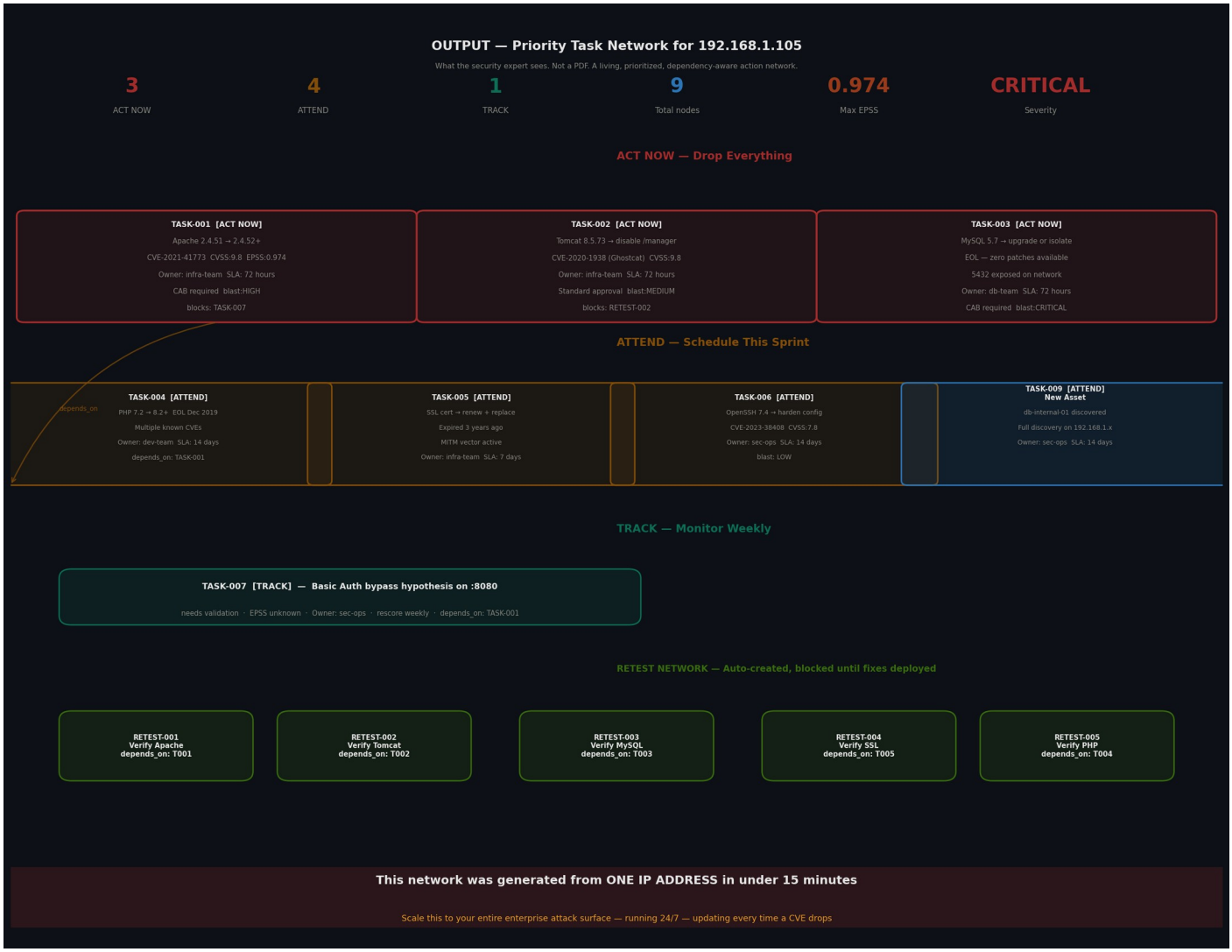

Figure 20 — The output: full priority task network from one IP address in 22 minutes

3 ACT NOW tasks · 4 ATTEND tasks · 5 auto-created retests · 1 new asset requiring discovery · From one IP address.

Every task has: an owner with name and team, a deadline derived from SSVC band plus org SLA configuration, a blast radius assessment for change management, a dependency chain showing what must complete first, and a pre-wired retest node that unblocks automatically

when the fix deploys. The network updates every night when EPSS scores refresh. If any CVE's probability changes by more than 5%, the associated task rescores automatically.

12. Why This Cannot Wait — The Real Numbers

Everything in this document is based on published 2025 and 2026 data. These are measurements, not projections.

45.4%	−1 day	137 days	$10.22 M
enterprise vulnerabilities unpatched after 12 months *Edgescan Vulnerability Statistics Report 2025*	**average time-to-exploit in 2025 attackers move before patches exist** *Mandiant M-Trends 2026 + Hadrian Security*	**average time to remediate a critical vulnerability** *Bitsight 2024 Remediation Analysis*	**average US data breach cost in 2025 — up 9% YoY** *IBM Cost of a Data Breach Report 2025*

IBM's 2025 data: organizations using extensive AI and automation paid $3.62M per breach on average versus $5.52M for those with no AI use. A $1.9M difference per incident — before accounting for frequency reduction from faster remediation. There are 4 million unfilled cybersecurity jobs globally. You cannot hire your way out of 133 new CVEs per day.

The RCTG does not replace security professionals. It replaces the part of the job that is triaging 10,000 scanner findings to find the 3 that matter this week. That is the part burning people out and causing 60% of KEV vulnerabilities to miss their remediation deadlines.

The question is whether you build this before the next breach happens.